TARTU ÜLIKOOL
Arvutiteaduse teaduskond
Informaatika õppekava

Rasmus Saks

Crumble kontrollerile Pythoni-laadse programmeerimiskeele loomine

Bakalaureustöö (9 EAP)

Juhendaja: Aivar Annamaa

Tartu 2017

Crumble kontrollerile Pythoni-laadse programmeerimiskeele loomine

Lühikokkuvõte: Crumble on Redfern Electronics'i loodud trükkplaat, millega on võimalik juhtida sellega ühendatud elektroonikakomponente. Crumble'it on võimalik programmeerida visuaalses programmeerimiskeeles erinevaid plokke kui pusletükke kokku vedades. Kuigi sellisel viisil programmeerimine on lihtsamini arusaadav algajatele, siis toetavad tekstilised programmeerimiskeeled nagu Python edasisi õpinguid paremini. Käesoleva töö eesmärgiks on luua Pythonil põhinev programmeerimiskeel Crumble kontrolleri juhtimiseks. Loodud keeles kirjutatud programme on võimalik arvutis kompileerida ning otse Crumble kontrollerisse läbi USB ühenduse saata.

Võtmesõnad: Crumble, Python, programmeerimiskeel, riistvara

CERCS: P170 (Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria))

Creating a Python-like Programming Language for the Crumble Controller

Abstract:

Crumble is a printed circuit board created by Redfern Electronics, which can be used to control electronics components attached to it. It is possible to program Crumble in a visual programming language by dragging together different blocks of code in a puzzle-like manner. Although programming in this way is easier to understand for novices, text-based programming languages such as Python are much more supportive towards a student's further studies in programming. The aim of this thesis is to create a programming language based on Python for the Crumble controller. One would be able to compile programs written in the language on their computer and send them directly over USB to the Crumble controller.

Keywords: Crumble, Python, programming language, hardware

CERCS: P170 (Computer science, numerical analysis, systems, control)

Sisukord

Sissejuhatus **5**

1 Crumble **6**
1.1 Redfern Electronics'i toodetud originaalkomponendid 6
1.1.1 Kontroller 6
1.1.2 Crumble-sõbralike komponentide komplekt 7
1.1.3 Ultraheli kaugusandur 8
1.1.4 Joonejälitaja 8
1.1.5 Sparkle 9
1.1.6 Crumbliser 9
1.1.7 Mootor 10
1.2 4tronix'i toodetud komponendid 10
1.2.1 Kiirendusandur 11
1.2.2 Lähedusandur 11
1.2.3 Puudutusandur 12
1.2.4 Numbrinäidik 12
1.2.5 Passiivne infrapunasensor 13
1.2.6 Servomootor 14
1.2.7 PlayGround 14
1.3 Arenduskeskkond ja -protsess 15
1.3.1 Sisend-väljundplokid 16
1.3.2 Sparkle'ite juhtimisplokid 17
1.3.3 Juhtplokid 18
1.3.4 Muutujaplokid 19
1.3.5 Tehteplokid 20
1.4 Arhitektuur 20

2 Crumblepy **22**
2.1 Arhitektuur 22
2.2 Kompilaator 23
2.3 Keele võimalused 25
2.3.1 Funktsioonid 25
2.3.2 Tingimuslaused 26
2.3.3 Tsüklid 27
2.3.4 Muutujad 27
2.4 Kompilaatori kasutamine 28
2.5 Testimine 29

Kokkuvõte **30**

Viidatud kirjandus **31**

Lisad **34**

1 Crumble kontrolleri masinkoodi käsud **34**

2 Litsents **35**

Sissejuhatus

Tänapäeval on tehnika lahutamatu osa inimeste eludest. Üha rohkem ja varajasemast east õpetatakse koolides lastele programmeerimist, kuid tihti jääb programmeerimise seos riistvaraga tähelepanuta. Selle probleemiga tegelemiseks on loodud mitmeid riistvarakomplekte, mida on võimalik küllaltki lihtsasti ka põhikooliõpilasel kokku panna ning arvuti abil programmeerida käituma vastavalt oma soovile. Üks sellistest komplektidest on firma Redfern Electronics loodud Crumble. Crumble võimaldab õpilastel, kes on programmeerimisega juba väheke tuttavad, lihtsa vaevaga ehitada ja programmeerida oma enda füüsiline robot, valgusfoor jne.

Crumble'i probleem seisneb aga liiga algelises ja lihtsas programmeerimiskeeles. Programmeerimiseks veetakse omavahel kokku klotse nagu pusletükke. Selline programmeerimine on algajale lihtne, kuid ei toeta tema edasist arengut, kuna reaalne programmeerimine toimub siiski tekstipõhiselt. Käesoleva töö eesmärgiks on luua Crumble'ile programmeerimiskeel, mis põhineb keelel Python. Kuna Python on paljudes koolides (sh Tartu Ülikoolis) õpilaste esimeseks programmeerimiskeeleks, siis on võimalik selle keele abil siduda Pythoni programmeerimise õppimine riistvara programmeerimisega. Samuti on võimalik käesolevat tööd kasutada õppevahendina Crumble'i süsteemiga tutvumiseks.

Töö on jaotatud kaheks peatükiks. Esimeses peatükis antakse ülevaade Crumble'ist ning sellega ühilduvatest komponentidest. Teises peatükis tutvustatakse töö raames loodud programmeerimiskeelt Crumblepy.

1 Crumble

Crumble on Redfern Electronics'i loodud trükkplaat, millega on võimalik juhtida nii spetsiaalselt Crumble'i jaoks ehitatud sisend-väljundkomponente (koodnimega Crumb'id) kui ka tavalisi elektroonikakomponente ning valgusdioode (vt ptk 1.1.5) [1]. Crumb'id erinevad analoogilistest elektroonikakomponentidest selle poolest, et neil on spetsiaalsed Crumble'ile mõeldud pesad. Nende pesade eelised on kirjeldatud peatükis 1.1.1

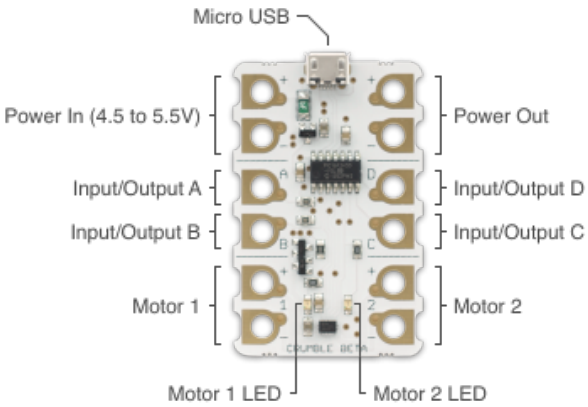
Järgnevad jaotused tutvustavad Crumble'i kontrollerit, komponente, mida sellega on võimalik ühendada, ning tarkvara, milles on võimalik luua programme kontrolleril jooksutamiseks.

1.1 Redfern Electronics'i toodetud originaalkomponendid

Crumble'ile loovad komponente põhiliselt kaks firmat: Redfern Electronics, kes on Crumble'i originaalautorid, ning 4tronix [2]. Selle peatüki alampeatükid räägivad olulisematest komponentidest, mida Redfern Electronics toodab ja müüb.

1.1.1 Kontroller

Kontroller on Crumble'i keskne komponent ning vajalik kõigi teiste komponentide tööks. Sellel on sisemälu ja protsessor, et salvestada jooksutatavat programmi ning juhtida ühendatud komponentide tööd. Kontrolleril on ühendused kahele mootorile, neljale sisend-väljundkomponendile, ühele energiasisendile (nt patareipakist), ühele energiaväljundile (nt valgusdioodile) ning Micro-USB kaablile [1]. Joonisel 1 on ülevaade Crumble kontrollerist.



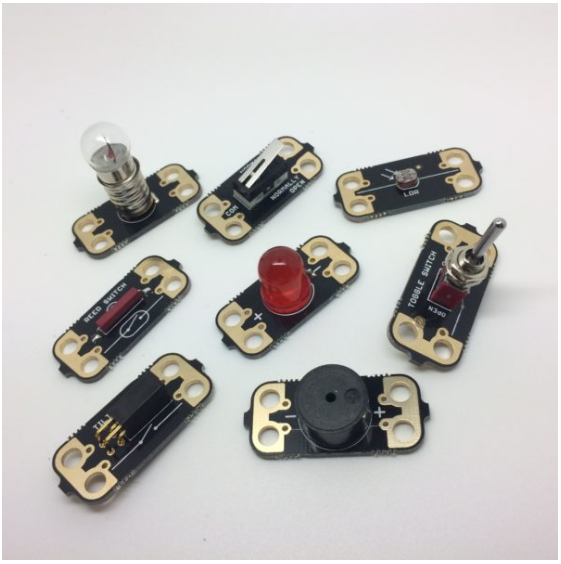
Joonis 1: Crumble'i kontrolleri ülevaade [3].

Iga kontrolleri pesa on loodud võimaldamaks mitut erinevat ühendusviisi: suur 4mm auk sobib mugavaks ühendamiseks krokodilljuhtmete, elektrit juhtiva niidi või terminalplokiga[1] ning suur juhtiv pind sobib juhtmete jootmiseks [1].

Kontrolleri saab ühendada Micro-USB pesa kaudu arvutiga, kust on võimalik sellele saata jooksutamiseks programme. Programmide loomiseks on Redfern Electronics loonud spetsiaalse tarkvara, mis kasutab MIT Scratchi-sarnast [4] kasutajaliidest. Tarkvarast kirjutatakse lähemalt peatükis 1.3.

Crumble kontroller võib toidet saada nii arvutist läbi USB ühenduse kui ka muust 4.5 - 5.5 V energiaallikast läbi energiasisendühenduse. Väliseks energiaallikaks sobivad muuhulgas standardsed AA suuruses patareid.

1.1.2 Crumble-sõbralike komponentide komplekt



Joonis 2: Crumble-sõbralikud komponendid [5].

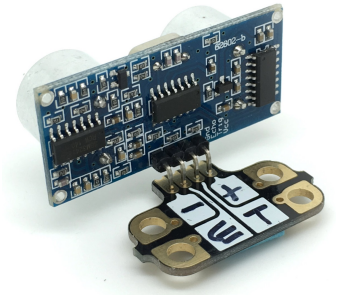
Redfern Electronics pakub komplekti kaheksa Crumble-sõbraliku komponendiga [6]. See sisaldab endas järgnevaid komponente (joonisel 2 alustades vasakult ülevalt ning liikudes paremale alla): lampi, mikrolülitit[2], fototakistit, keelreleed[3], valgusdioodi, tumblerlülitit, kallutusandur-lülitit ja sumistit. Crumble-sõbralikuks teeb need komponendid asjaolu, et neil on Crumble'iga mugavamaks ühendumiseks spetsiaalsed pesad.

[1] vt ptk 1.2.7

[2] kahe väljundi vahel lülitatav lüliti [7]

[3] magnetväljale reageeriv lüliti [8]

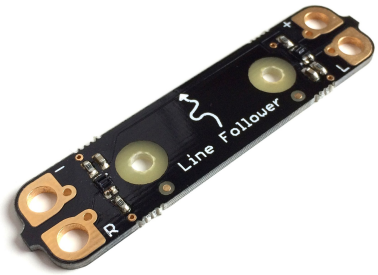
1.1.3 Ultraheli kaugusandur



Joonis 3: Ultraheli kaugusanduri tagakülg [9].

Ultraheli kaugusandur [10] võimaldab Crumble programmil mõõta anduri kaugust seinast. Väiksemate objektide kauguse mõõtmiseks ei ole antud kaugusandur sobilik, kuna see vajab ultrahelisignaali tagasipõrkamiseks suurt tasast pinda (nagu sein). Tegemist on HC-SR04 [11] kaugusanduriga, millele on Crumbliser'iga (vt ptk 1.1.6) lisatud Crumble ühendused. Alates tarkvara versioonist 0.25.0 on võimalik seda komponenti juhtida ühe käsuplokiga [10]. Joonisel 3 on näha anduri tagakülg koos sellega ühendatud Crumbliser'iga.

1.1.4 Joonejälitaja

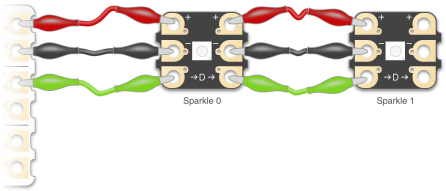


Joonis 4: Joonejälitaja [12].

Joonejälitaja võimaldab Crumble'i põhjal ehitatud robotil sõita mööda maha märgitud joont. Jälitaja väljastab kaht signaali: üks signaal, kui on tarvis vasakule pöörata, ning teine signaal, kui on tarvis paremale pöörata. Joonisel 4 on näidatud

joonejälitaja. +/- ühendused on toitevoolu jaoks ning L ja R väljundid vastavalt vasakule ja paremale pööramiseks.

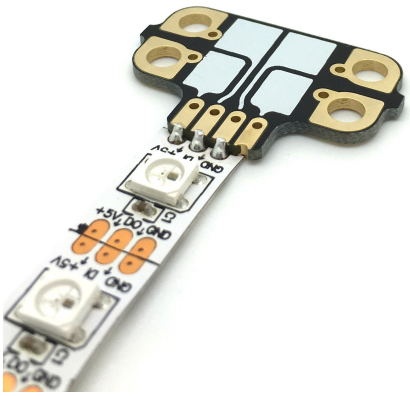
1.1.5 Sparkle



Joonis 5: Sparkle'ite ühendamine Crumble kontrolleriga [13].

Sparkle'iteks [14] nimetatakse Crumble'iga ühilduvaid täisvärvides valgusdioode. Need on standardiga WS2812B [15] ühilduvad, mis tähendab, et Crumble'iga on võimalik kõiki selle standardi valgusdioode ühendada ning neid individuaalselt juhtida. Erinevalt Sparkle'itest pole selliseid valgusdioode üldjuhul nii lihtne Crumble'iga ühendada, sest neil puuduvad spetsiaalsed pesad juhtmete hoidmiseks. Joonisel 5 on näidatud, kuidas on võimalik mitu Sparkle'it jadamisi ühendada. Crumble tarkvara võimaldab kuni 32 erinevat Sparkle'it individuaalselt juhtida.

1.1.6 Crumbliser



Joonis 6: Crumbliseriga ühendatud WS2812B tüüpi valgusdioodriba [16].

Crumbliser [17] on Crumble komponent, mis võimaldab suvalistele elektroonikakomponentidele külge joota Crumble ühendused. Selleks on Crumbliser plaadil

neli väiksemat jootmiseks mõeldud ühendust, millest igaüks on ühendatud suurema Crumble-tüüpi ühendusega. Iga ühenduse kõrval on ka valge ala, kuhu saab markeriga meeldetuletuseks kirjutada iga ühenduse otstarbe. Joonisel 6 on kujutatud WS2812B tüüpi valgusdioodriba, millele on Crumbliseri abil lisatud Crumble-tüüpi ühendused.

1.1.7 Mootor

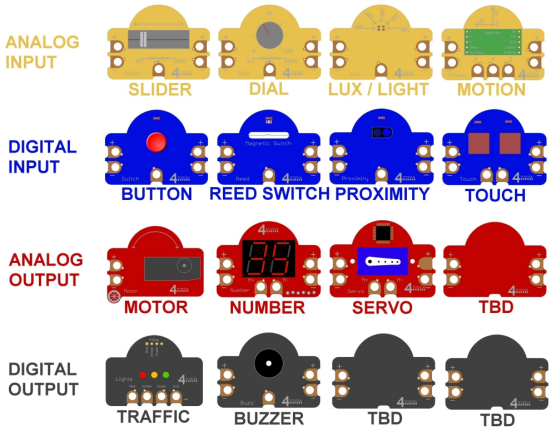
Joonis 7: Kaks mootorit [18].

Redfern Electronics toodab ka Crumble'iga ühilduvaid mootoreid [19]. Mootor tuleb ühendada kontrolleril ühte kahest spetsiaalselt mootorite ühendamiseks mõeldud pessa. Joonisel 7 on kujutatud kaks mootorit, millel on krokodilljuhtmed, et Crumble'iga lihtsasti ühenduda.

1.2 4tronix'i toodetud komponendid

Lisaks Redfern Electronics'ile toodab Crumble'ile komponente ka Ühendkuningriigist pärit firma 4tronix [20].

Joonisel 8 on kujutatud valik 4tronix'i komponentidest Crumble'ile. Kuigi mõned nendest kattuvad Redfern'i enda poolt toodetud komponentidega, siis nendel on oluline disainierinevus. Peaaegu kõigil firma 4tronix toodetud Crumble'ile mõeldud komponentidel on kaks +/- ühenduspaari ning eraldi ühendus(ed) komponendi väljundi(te) jaoks. See võimaldab nende Crumb'e omavahel jadamisi ühendada ning sellega laiendada kontrolleri energiaväljundpesasid. Samuti on need komponendid värvi järgi kategoriseeritud. Kollased komponendid on analoogsisendkomponendid, sinised digitaalsisendkomponendid, punased analoogväljundkomponendid ning mustad digitaalväljundkomponendid. Järgnevalt on välja toodud olulise-

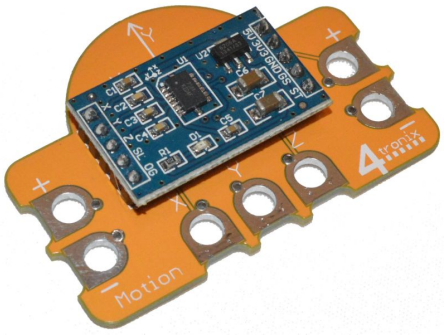


Joonis 8: Firma 4tronix toodetud Crumb'id [21].

mad komponendid, mida 4tronix toodab ning mida Redfern ei tooda.

1.2.1 Kiirendusandur

Kiirendusandur on seade, millega on võimalik mõõta kiiruse muutumist (kiirendust) [22]. Selleks on kiirendusanduri Crumb'il kolm väljundit kolmes dimensioonis kiirenduse lugemiseks: X, Y ja Z [23].

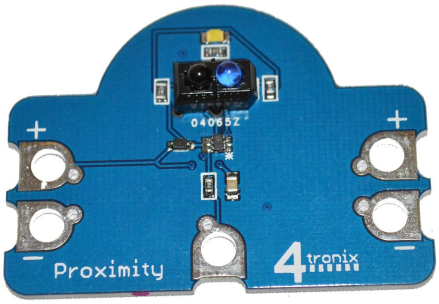


Joonis 9: 4tronix Crumble'ile loodud kiirendusandur [24].

Joonisel 9 on 4tronix'i toodetud kiirendusandur. Selle komponendi plaadi ülalosas on ka toodud joonis näitamaks, mis suunas X, Y ja Z väljundid kiirendust väljastavad.

1.2.2 Lähedusandur

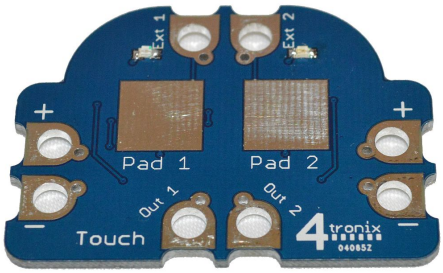
Lähedusandur (joonisel 10) võimaldab tuvastada, kas mingi objekt on anduri lähedal. Komponendil on selle tarbeks infrapunaemitter ja -vastuvõtja, et mõõta



Joonis 10: 4tronix lähedusandur [25].

lähedal asuva objekti pealt peegelduva infrapunakiirguse intensiivsust ning seeläbi hinnata, kas tema ees on mingi objekt [26]. Kui komponent tuvastab, et tema ees asub lähedal objekt, siis süttib valgusdiood komponendi peal ning väljastatakse digitaalne signaal 1. Anduri tuvastamiskaugus on umbes 2-10 cm olenevalt objektist.

1.2.3 Puudutusandur



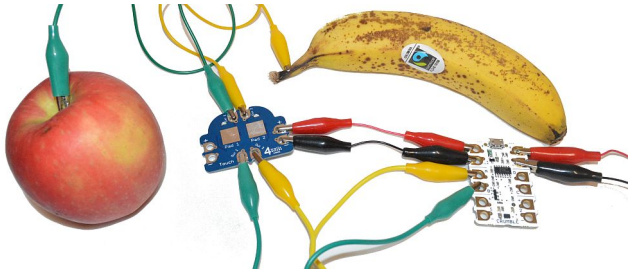
Joonis 11: 4tronix puudutusandur [27].

Puudutusanduril (joonis 11) on kaks ala, mille puudutamisel süttib indikaatortuluke ning väljastatakse vastavast väljundist digitaalne signaal 1. Samuti on plaadil ka kaks lisaühendust (märgitud „Ext 1" ja „Ext 2"), millega on võimalik laiendada vastavaid puudutusalasid teiste puudutusalade või objektidega (nt puuviljad). Joonisel 12 on näidatud, kuidas seda on võimalik kasutada.

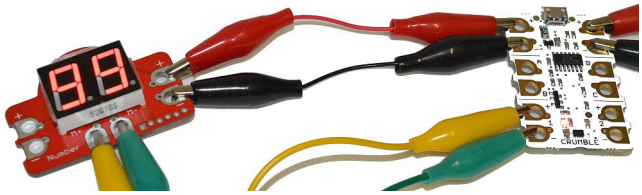
Kuigi sellisel puuviljadega ühendamisel on vähe praktilist kasutust, siis on see siiski huvitav ja ebatavaline viis oma projekti juhtimiseks.

1.2.4 Numbrinäidik

Numbrinäidik on analoogväljundseade, mis ühendub kontrolleri mootoripesadesse ning väljastab 7-segmendilistel näidikutel numbrit -100-st +100-ni vastavalt moo-



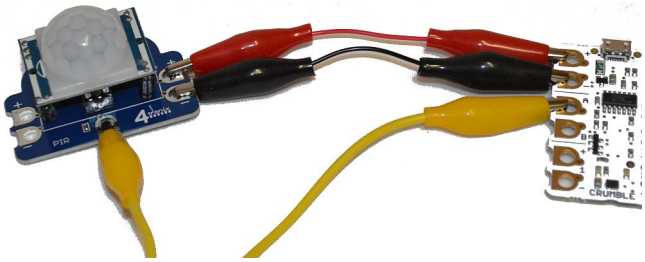
Joonis 12: Puuviljadega ühendatud puudutusandur [28].



Joonis 13: Crumble kontrolleriga ühendatud numbrinäidik [29].

toripesa väljastatud väärtusele [30]. Negatiivsete arvude kuvamisel süttivad lisaks vastava positiivse numbri kuvamiseks tarvilikud segmendid ka kuvari allääres olevad kaks täppi. Arvu 100 (ja -100) kuvamiseks on kasutatud omapärast lahendust, sest kasutada on vaid kaks numbrinäidikut. Selleks süttivad vasakpoolse näidiku ülemine horisontaalne segment (moodustades horisontaalse numbri 1) ning mõlemal näidikul alumised neli segmenti (moodustades kumbki numbri 0). Negatiivse arvu 100 puhul süttivad ka täpid näidiku allääres. Suuremate arvude kui 100 ja väiksemate arvude kui -100 kuvamise reeglid pole üheselt määratud. Joonisel 13 on kujutatud numbrit 99 kuvav numbrinäidik.

1.2.5 Passiivne infrapunasensor

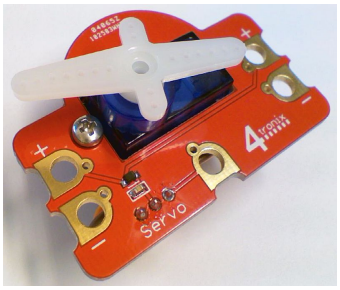


Joonis 14: Crumble kontrolleriga ühendatud passiivne infrapunasensor [31].

Passiivne infrapunasensor (joonisel 14) mõõdab muutusi ümbritsevas infrapunakiirguses ning võimaldab sellega tuvastada muuhulgas inimeste liikumist [32].

Liikumise tuvastamisel väljastab sensor digitaalse signaali 1.

1.2.6 Servomootor

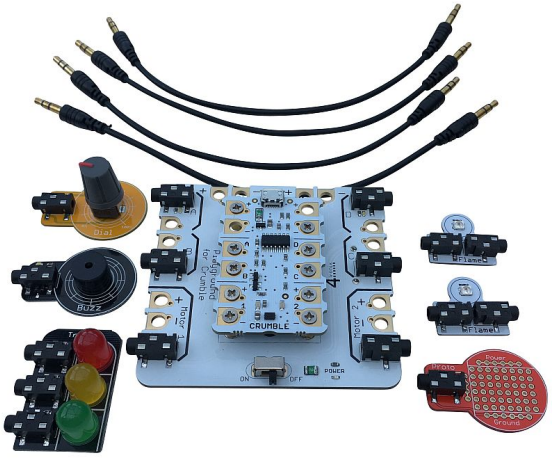


Joonis 15: 4tronix servomootor [33].

Servomootor erineb tavalisest mootorist selle poolest, et kui signaali saamisel tavaline mootor pöörleb pidevalt, siis servomootor pöörab end vastavalt sisendsignaalile kindlasse asendisse ja üritab seda asendit säilitada [34]. 4tronix servomootor (joonisel 15) võimaldab võtta asendi -90 ja 90 kraadi vahel [35].

1.2.7 PlayGround

PlayGround (joonis 16) on 4tronix'i loodud terminalplokk selleks, et Crumble (ja Micro:Bit) komplekte mugavamalt kasutada [36]. Crumble kontroller ühendub kruvidega PlayGround plaadi külge. Plaadi all on kontrolleri toiteks patereipakk ning külgedel erinevad sisend-väljundpesad.



Joonis 16: PlayGround, Gizmod ja juhtmed [37].

4tronix kirjutab oma blogisissekandes [36], et PlayGroundi teeb eriliseks see, et lisaks tavalistele Crumble-tüüpi ühendustele on nad lisanud ka 3,5-millimeetrised pesad, mis on paremini tuntud ka kui pesad kõrvaklapijuhtmete ühendamiseks. Nendes pesades on ühte kokku toodud kolm Crumble-tüüpi pesa: voolu sisend-väljund ning signaal. See võimaldab projekte oluliselt paremini organiseerida, sest tavalise kolme juhtme asemel iga Crumb'i kohta on võimalik sama tulemust saavutada ühe juhtmega ning kõrvaklapijuhtmete ühendamine om palju lihtsam kui krokodilljuhtmete ühendamine.

4tronix on loonud paljudest juba nende toodetud Crumb'idest spetsiaalselt PlayGroundile mõeldud versioonid (Gizmod), millel on Crumble-tüüpi pesade asemel 3,5-millimeetrised pesad PlayGroundiga (või teiste Gizmodega) ühendumiseks. Lisaks pakuvad nad ka prototüüpimise Gizmot (joonisel all paremal), millega on võimalik ise luua PlayGroundiga ühenduv Gizmo [38].

1.3 Arenduskeskkond ja -protsess

Crumble'ile programmide loomiseks on tarvis mikrokontroller USB kaudu arvutiga ühendada. Samuti on vajalik alla laadida ja arvutisse paigaldada Crumble tarkvara[4].

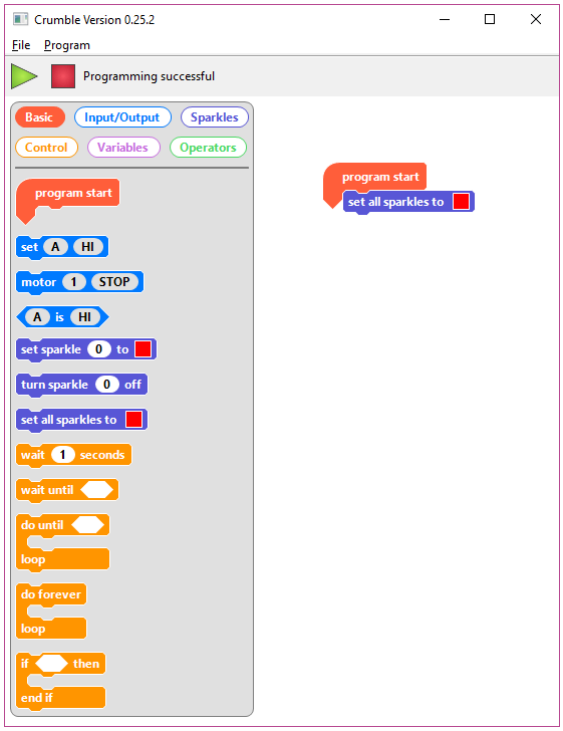
Crumble tarkvara on MIT Scratchist inspireeritud keskkond Crumble'i programmide arendamiseks [1]. Sarnaselt Scratchile luuakse programme erinevate plokkide kokkuvedamisega. Iga plokk on vastavalt oma tüübile kindla kujuga ning plokkidel võivad olla ka kindla kujuga augud, kuhu saab teisi plokke sisestada. Kõik käsud on pusletükikujulised ning sobituvad üksteise järele, tõeväärtused on kuusnurgad ning arvud ja muutujad ümardatud otstega ristkülikud.

Joonisel 17 on näidatud Crumble tarkvara koos lihtsa programmiga, mis seab kõik Sparkle'id punaseks. Vasakul ääres on valik plokke erinevates kategooriates: „Basic" (enimkasutatavad), „Input/Output" (sisend-väljnud), „Sparkles" (Sparkle'id), „Control" (programmi juhtplokid), „Variables" (muutujad) ja „Operators" (tehted).

Tarkvaral on kaks nuppu: roheline kolmnurk ja punane ruut. Roheline nupp laadib praeguse programmi USB ühenduse kaudu Crumble kontrollerisse ning käivitab selle. Punane nupp peatab hetkel jooksva programmi. Samuti on võimalik rippmenüüst „File" Crumble programme arvutisse salvestada ja sealt uuesti laadida.

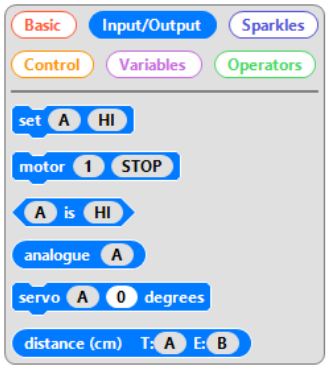
Iga programm peab koosnema täpselt ühest „Program start" plokist. See plokk tähistab programmi alguse kohta, seega kogu programmi tegevus algab sealt. Järgnevalt on toodud kategooriate kaupa kõikide plokkide kirjeldused.

[4]Kättesaadav aadressil `http://redfernelectronics.co.uk/crumble-software/`



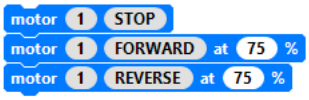
Joonis 17: Crumble tarkvara.

1.3.1 Sisend-väljundplokid



Joonis 18: Sisend-väljundplokkide nimekiri.

Sisend-väljundplokid (joonisel 18) võimaldavad digitaalselt ja analoogiliselt lugeda ja digitaalselt kirjutada kontrolleri A, B, C ja D pesadele. Kirjutamiseks on „set" plokk, mis võimaldab valitud pesalt väljastada digitaalse signaali 1 või 0 (HI või LO).



Joonis 19: „Motor” ploki kolm seadistust

„Motor” plokk juhib mootoripesasid. Võimalik on valida pesa 1 või 2 vahel ning kas mootor peatada, edasi liikuda või tagasi liikuda (vastavalt „STOP”, „FORWARD” ja „REVERSE”, vt joonis 19) ning kui kiiresti protsendina 0-st 100-ni.

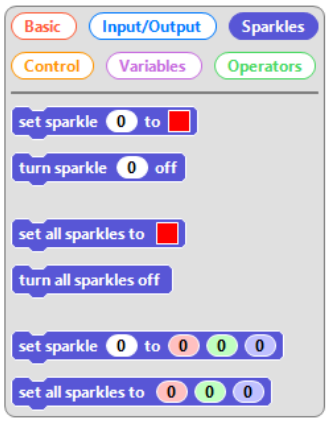
„Servo” plokiga on võimalik seada kindel servomootor (vt 1.2.6) mingisse teatud asendisse. Tuleks tähele panna, et servomootor ei ühendu mootoripesadega vaid väljundpesaga A, B, C või D (lisaks energiaväljundile).

„(A/B/C/D) is (HI/LO)” plokk tagastab, kas valitud sisendpesast tulev signaal on sees või mitte. Näiteks „A is HI” plokk tagastaks tõese väärtuse, kui pesaga A ühendatud nuppu all hoitakse.

„Analogue (A/B/C/D)” plokki kasutatakse analoogsignaali saamiseks valitud pesast. See on kasulik, kui Crumb'i tagastatud väärtus on pidev suurus ning mitte kas 0 või 1. Analoogsignaal kajastub koodis arvuna 0-st 255-ni. Seda kasutatakse näiteks kiirendusanduri (vt 1.2.1) puhul.

„Distance” plokk on ultraheli kaugusandurilt (vt 1.1.3) tema vastas oleva objekti kauguse teada saamiseks. Selleks tuleb anduri T ja E ühendused ühendada kontrolleriga ning „distance” plokis ära määrata. Plokk tagastab andurilt saadud kauguse sentimeetrites.

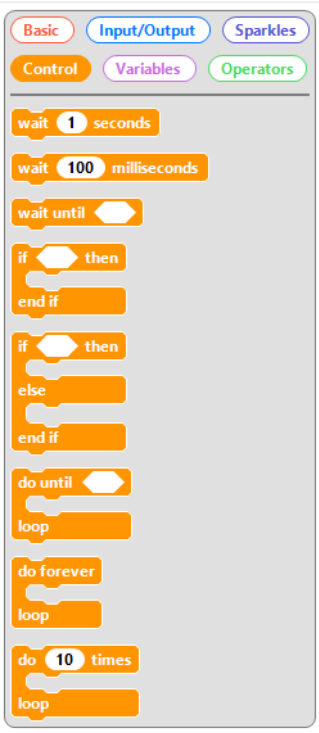
1.3.2 Sparkle'ite juhtimisplokid



Joonis 20: Sparkle'ite juhtimisplokkide nimekiri.

Kõiki Sparkle'eid on võimalik nii korraga kui ka individuaalselt muuta kindlaks värviks. Selleks on plokid „set sparkle to" ning „set all sparkles to". Esimene seab kindla Sparkle'i kindlaks värviks ning teine teeb sama kõikide Sparkle'itega. Nendest plokkidest on Crumble tarkvaras kaks varianti: värvivalimisdialoogiga variant (joonisel 20 esimene ja kolmas plokk), mis võimaldavad kasutajal spetsiaalse akna abil valida sobiv värv, ning RGB väärtustega variant (viies ja kuues plokk joonisel), kus kasutaja saab määrata punase, rohelise ja sinise värvi intensiivsuste (vahemikus 0 kuni 255) kombinatsiooniga Sparkle'i värvi. Plokid „turn sparkle off" ja „turn all sparkles off" lülitavad vastavalt ühe kindla või kõik Sparkle'id välja.

1.3.3 Juhtplokid

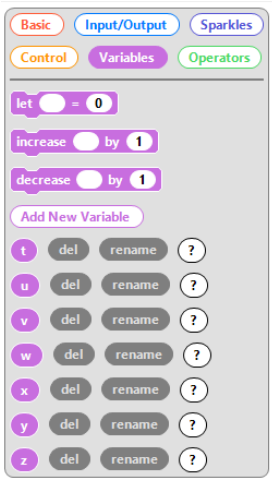


Joonis 21: Juhtplokkide nimekiri.

Juhtplokid (joonisel 21) võimaldavad programmi tööd juhtida vastavalt teatud tingimustele. „Wait seconds" ja „wait milliseconds" plokid ootavad vastavalt mingi arvu sekundeid või millisekundeid enne järgmise käsu täitmist. „Wait until" plokk peatab programmi töö nii kaua kuni teatud tingimus on täidetud. „If then" ning „If then else" plokid võimaldavad mingi tingimuse kohaselt käivitada või mitte käivitada koodiplokke. „Do until", „do X times" ja „do forever" plokid võimaldavad

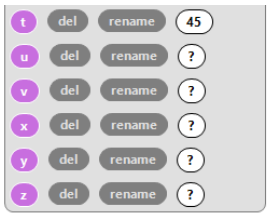
koodiplokke korduvalt käivitada vastavalt mingi tingimuse täitumiseni, täpselt X korda või lõpmatu arv kordi.

1.3.4 Muutujaplokid



Joonis 22: Muutujaplokkide nimekiri.

Muutujaplokid (joonisel 22) võimaldavad luua muutujaid, seada neile väärtuseid ning nende väärtuseid lugeda. „Let" plokk seab muutujale väärtuse. Muutuja plokk tuleb tõmmata „Add New Variable" nupu all olevatest muutujaplokkidest. Algselt on olemas muutujaplokid nimedega „t", „u", „v", „w", „x", „y" ja „z", kuid neid on võimalik kustutada (nupuga „del"), ümber nimetada (nupuga „rename") ning juurde lisada (nupuga „Add New Variable"). „Increase by" plokk lisab muutuja väärtusele mingi arvu. „Decrease by" plokk vähendab muutuja väärtust mingi arvu võrra.

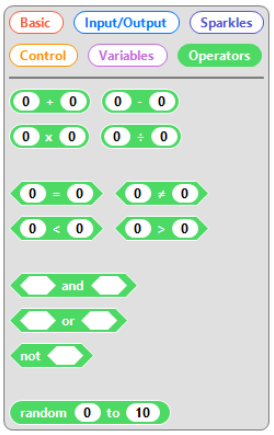


Joonis 23: Muutuja „t" väärtus on 45.

Programmi jooksmise ajal on võimalik kõigi muutujate väärtuseid muutujaplokkide nimekirjas näha. Joonisel 23 on näha, et muutuja „t" väärtuseks on 45.

Teiste muutujate väärtused on „?”, sest nendele pole programmi töös määratud väärtust.

1.3.5 Tehteplokid



Joonis 24: Tehteplokkide nimekiri.

Tehteplokid (joonis 24) võimaldavad teha programmi töös esinevate väärtustega aritmeetilisi ja loogilisi operatsioone. Aritmeetikast on võimaldatud liitmine, lahutamine, korrutamine ja jagamine, kusjuures jagamine on täisarvuline jagamine, kus vastuseks on vaid täisosa ning jääk visatakse minema. See on põhjustatud sellest, et Crumble ei võimalda ujukomaarvudega töötamist. Loogilistest operatsioonidest on võimaldatud võrdus, mittevõrdus, „suurem kui”, „väiksem kui”, „ja”, „või” ning „mitte”. Lisaks on tehteplokina toodud ka „random” plokk, mis tagastab suvalise täisarvu etteantud vahemikus (mõlemad otspunktid kaasaarvatud).

1.4 Arhitektuur

Crumble kontroller kasutab oma protsessoriks PIC16F1455[5] tüüpi mikrokontrollerit. Sellel on 14 KB mälu programmide salvestamiseks ning 1024 baiti sisemälu [39]. Komponentide juhtimiseks on Redfern Electronics loonud kontrollerile oma masinkoodi. Katse-eksitus meetodil ning Crumble tarkvara lähtekoodi lugedes leitud masinkoodi käsud ja nende kirjeldused on toodud lisas 1.

Käsu enda argumendid (näiteks käsu „`POP` a” puhul a) kodeeritakse koos käsu koodiga otse masinkoodi sisse. Käsu kood on tabelis esitatud kuueteistkümnendsüsteemi arvuna grupeeritud baitide kaupa. See on käsu numbriline esitus, mida Crumble protsessor lugeda ja millele vastavalt käituda oskab. Käsu argumendi

[5] `http://www.microchip.com/wwwproducts/en/PIC16F1455`

kodeerimiseks lühendatakse see ühe baidi (kümnendüsteemi arv vahemikus 0 kuni 255) pikkuseks ning liidetakse otse käsu koodile. Näiteks käsu „`POP` 45" masinkoodi esitusviis oleks „`03 2D`", sest „`POP`" masinkoodis on `03 00` (vt lisa 1), 45 kuueteistkümnendsüsteemis on `2D` ning `03 00` + `2D` = `03 2D`. Samuti on paljudel käskudel olemas ka pinu argumendid. Pinu argumendid salvestatakse programmi töö käigus protsessori pinusse, mille pealt käsud oma argumente loevad ning tagastusväärtuse olemasolul ka selle pinusse kirjutavad. Loogikatehete puhul (näiteks „`OR`" või „`AND`") loetakse tõeseks väärtuseks kõike, mis ei ole 0.

Lisas 1 toodud käsud on madalaima taseme ehituskivid Crumble'ile programmide loomiseks. Nende kombineerimisel kõrgema taseme funktsionaalsusteks põhineb nii Crumble tarkvara kui ka järgmises peatükis kirjeldatud ning töö eesmärgina loodav programmeerimiskeel Crumblepy.

2 Crumblepy

Crumblepy[6] on käesoleva töö raames loodud Pythonil põhinev programmeerimiskeel, millega on võimalik juhtida Crumble kontrolleri tööd. Järgnevad peatükid kirjeldavad nii Crumblepy ehitust ja mis valikuid selleni jõudmiseks tehti kui ka tööprotsessi sellega töötamisel.

2.1 Arhitektuur

Kuigi Crumble tarkvara on loodud Python 2-s, siis Crumblepy on kirjutatud Python 3-s. See erinevus tuleneb põhiliselt sellest, et Pythoni loojad ise soovitavad uute projektide alustamisel kasutada Pythoni kolmandat versiooni, sest see on paljudes aspektides parem kui Python 2 [40]. Lisaks sellele on autor tuttavam Python 3-ga kui Python 2-ga. Kuna Crumblepy põhineb Crumble tarkvara lähtekoodil, siis on tarvis Python 2-s kirjutatud kood ümber teisendada vastavaks Python 3 koodiks.

Failis „crumblepy/usb.py" on kokku pandud Redfern Electronics'i loodud Pythoni kood Crumble'iga suhtlemiseks USB kaudu, mis on ümber teisendatud Python 3-e. Antud juhul oli tarvis teha vähe muudatusi, et algne kood jookseks sama moodi Python 3-l, kui ta jooksis Python 2-l. Põhilised erinevused tulenesid täisarvude jagamisel[7] ning generaatorobjektide või listide tagastamisel[8].

Crumblepy on loodud olema algajatele võimalikult lihtne viis riistvaraprogrammeerimisega tegelemiseks. Seetõttu valiti Crumblepy keele põhjaks just Python, mida tihti õpetatakse koolides esimese programmeerimiskeelena. Samuti teeb Pythoni standardteek Pythoni parsimise väga lihtsaks, pakkudes selleks `ast` moodulit [42]. Mooduli funktsioon `ast.parse()` võimaldab parsida sisendsõne Pythoni abstrakseks süntaksipuuks nagu on näidatud joonisel 25.

```
import ast
with open("file.crpy") as f:
    syntaxtree = ast.parse("\n".join(f.readlines()))
```

Joonis 25: Faili parsimine abstraktseks süntaksipuuks

Tulemuseks on puu, mille tipud on Pythoni süntaksielemendid, mis on täieli-

[6] Kättesaadav aadressil `https://github.com/rasmussaks/crumblepy`

[7] Kahe täisarvu jagamine Python 2-s tagastab täisarvu ($5/2 = 2$), Python 3-s aga ujukomaarvu ($5/2 = 2.5$) [41]

[8] Python 2-s tagastasid paljud sisseehitatud funktsioonid (nt `range()` ja `dict().keys()`) list-tüüpi objekte, kuid Python 3-s tagastavad need generaatorobjekte [41]

kult kirjeldatud Pythoni dokumentatsiooni peatükis 32.2.2[9]. Selle puu rekursiivse läbimisega on võimalik kompileerida Pythonil põhinev sisendkood Crumble kontrollerile mõistetavaks masinkoodiks.

Crumble'i masinkoodi piirangute tõttu ei ole võimalik kõiki Pythoni keele võimalusi kompileerida, mistõttu on paljud nendest välja jäetud ning annavad kompileerimisel veateate. Sinna kuuluvad näiteks ujukomaarvud, andmetüübid `list` ja `dict` ning palju muud. Crumblepy keele võimalustest on kirjutatud peatükis 2.3.

2.2 Kompilaator

Crumblepy kompilaator (failis „crumblepy/compiler.py") kasutab Pythoni `ast` moodulit, et saada sisendfaili abstraktne süntaksipuu ning seejärel muundab selle Crumble kontrollerile arusaadavaks masinkoodiks. Vahelüliks masinkoodi ja Crumblepy koodi vahel on assemblerkood, mis on inimloetav masinkood. Joonisel 26 on toodud Crumblepy kood, mis väljastab signaali 1 pesast D, ning joonisel 27 on sama kood esitatud assemblerkoodis, kuhu on lisatud kommentaarid iga rea ülesande kohta.

```
set_output(D, 1)
```

Joonis 26: Python kood väljundpesa D sisselülitamiseks

```
PUSHL 2 ; Pesa number pinusse
PUSHL 1 ; Signaal 1 pinusse
DWR ; Võta pinust kaks arvu ning väljasta antud pesast antud signaal
STOP ; Programmi lõpp
```

Joonis 27: Kommenteeritud assemblerkood väljundpesa D sisselülitamiseks

Erinevalt lähtekoodist, kompileeritakse iga assemblerkoodi käsk üheks (välja arvatud mõnel erandjuhul kaheks) masinkoodi käsuks. Joonisel 26 on üks lähtekoodi käsk `set_output(D, 1)`, mis masinkoodis esitatakse kolme käsuga: kaks käsku argumentide pinusse panemiseks ning üks käsk väljundpesa väljundi seadmiseks vastavalt pinust loetud väärtustele.

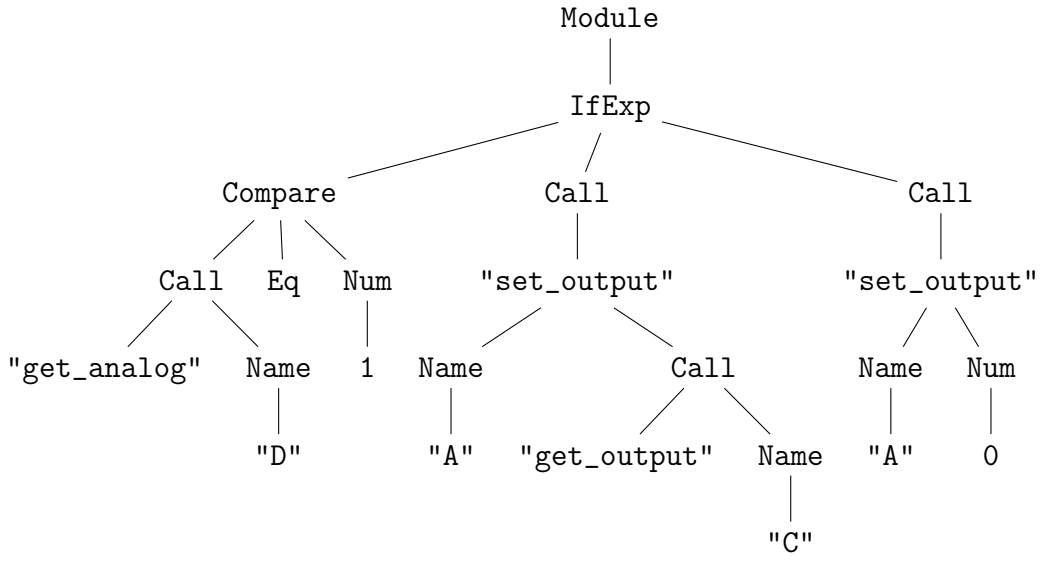
Joonisel 28 on toodud kood, mis seab pesa A väljundiks pesa C signaali, kui pesa D signaal on 1, või signaali 0, kui pesa D signaal ei ole 1.

Joonisel 29 on kujutatud joonisel 28 toodud Crumblepy koodi abstraktne süntaksipuu, mis on saadud funktsiooni `ast.parse()` kasutamisel antud koodi peal.

[9] `https://docs.python.org/3/library/ast.html#abstract-grammar`

```
if get_digital(D) == 1:
    set_output(A, get_digital(C))
else:
    set_output(A, 0)
```

Joonis 28: Crumblepy kood pesa A väljundi seadmiseks vastavalt pesa D signaalile



Joonis 29: Abstraktne süntaksipuu

Abstraktne süntaksipuu on koodi süntaktilise struktuuri esitus puuna. See võimaldab programmiliselt väikese vaevaga kompileerida Crumblepy lähtekood Crumble kontrolleri masinkoodiks. Joonisel 30 on väljavõte Crumblepy kompilaatori tuumast, rekursiivsest abstraktse süntaksipuu läbijast. Iga Pythoni abstraktse süntaksipuu juurtipuks on `Module`, millest hargnevad alamtipud vastavalt programmi sisule.

Funktsioon `compile_stmt` võtab argumendina sisse abstraktse süntaksipuu juurtippu (`Module` tüüpi tipu) ning läbib rekursiivselt iga puu tipu ning väljastab iga tipu kohta sellele vastava(d) masinkoodi käsu(d). Joonisel 31 on toodud Crumblepy funktsiooni `set_output(pesa, signaal)` kompileeriv funktsioon.

Funktsioon `self.compile_expr(node)` kompileerib antud avaldise tipu sellele vastavaks masinkoodiks. Muutuja `self.assembly` on viide `crumblepy.Assembly` klassile, kus on abifunktsioonid lisas 1 esitatud masinkoodikäskudega töötamiseks. Näidiskoodis joonisel 31 väljastab `self.assembly.dwr()` masinkoodi käsu `DWR`, mis loeb kahe eelneva `self.compile_expr(node)` käsu poolt pinusse pandud argumendid - pesa number ja signaali väärtus - ning väljastab antud pesast määratud

```
class Compiler:
    def compile_stmt(self, node):
        if type(node) == Module:          # Juurtipp
            for n in node.body:           # Läbi kõik juurtipu alamtipud
                self.compile_stmt(n)      # Rekursioon
        elif type(node) == If:            # if tipp
            self.if_stmt(node)            # Kompileeri if-lause
        # ...
```

Joonis 30: Rekursiivne süntaksipuu läbimine

```
def set_output(self, output_node, signal_node):
    self.compile_expr(output_node)
    self.compile_expr(signal_node)
    self.assembly.dwr()
```

Joonis 31: Crumblepy funktsiooni `set_output(pesa, signaal)` kompileeriv funktsioon

signaali.

Erinevalt Crumble tarkvarast puuduvad Crumblepy'l argumentidele ranged piirangud, mis võimaldavad Crumblepy keeles väljendada palju keerukamaid programme kui Crumble tarkvaras. Crumble tarkvara piirab näiteks funktsiooni „set" argumendid olema täpselt „A", „B", „C" või „D" ning „HI" või „LO". Crumblepy selliseid piiranguid ei sea, mis tähendab, et joonisel 28 olev kood on Crumblepy's lubatud, kuid Crumble tarkvaras mitte, sest seal kasutatakse ühe pesa väärtust teise pesa sisendina, mida Crumble tarkvaras ei ole võimalik teha.

2.3 Keele võimalused

Järgnevates peatükkides on välja toodud kõik funktsionaalsused, mida Crumblepy keel toetab. Kuigi keel põhineb Pythonil, ei ole võimalik kõiki Pythoni võimalusi Crumble kontrolleri masinkoodis väljendada, mistõttu on esindatud vaid kindel alamhulk Pythoni funktsionaalsusest.

2.3.1 Funktsioonid

Crumblepy's on võimalik kasutada erinevaid funktsioone Crumble'i juhtimiseks. Kõik funktsioonid käituvad identselt vastavatele funktsioonidele Crumble tarkvaras ning on esitatud tabelis 1.

Tabel 1: Crumblepy funktsioonid

Funktsioon	Kirjeldus
`set_output(pad, s)`	Väljastab pesast `pad` digitaalse signaali `s`.
`get_digital(pad)`	Loeb ja tagastab digitaalse signaali pesast `pad`.
`get_analog(pad)`	Loeb ja tagastab analoogsignaali pesast `pad`.
`wait(msec)`	Peatab programmi `msec` millisekundiks.
`random(a, b)`	Tagastab suvalise täisarvu `a` ja `b` vahel (mõlemad kaasa arvatud).
`set_motor_1(pct)`	Seab mootori 1 kiiruseks `pct`%, `pct` võib olla vahemikus −100 kuni 100.
`set_motor_2(pct)`	Seab mootori 2 kiiruseks `pct`%, `pct` võib olla vahemikus −100 kuni 100.
`set_servo(pad, d)`	Seab servomootori, mis on ühendatud pesaga `pad` nurgaks `d` kraadi.
`get_distance(e, t)`	Tagastab ultraheli kaugusanduri kauguse (vt ptk 1.1.3), kui anduri pesa „E” on ühendatud Crumble'i pesaga `e` ning anduri pesa „T” Crumble'i pesaga `t`

Pesade identifikaatoritena võib kasutada nii numbreid 0, 1, 2, 3 (vastavalt pesad „A”, „B”, „D”, „C”) kui ka sisse ehitatud muutujaid `A`, `B`, `C`, `D`.

2.3.2 Tingimuslaused

Crumblepy toetab täielikult Pythoni tingimuslauseid (**`if`**-lauseid). Joonisel 32 on toodud näide **`if-else`** konstruktsioonist ning joonisel 33 on toodud sama koodi esitus masinkoodis koos selgitavate kommentaaridega.

```
if get_digital(D) and not get_digital(A):
    set_output(C, 1)
else:
    set_output(C, 0)
```

Joonis 32: **`if-else`** konstruktsioon Crumblepy koodina

```
PUSHL 2          ; Pesa D
DRD              ; Loe pesast D digitaalne signaal
PUSHL 0          ; Pesa A
DRD              ; Loe pesast A digitaalne signaal
NOT_OP           ; Pesa A vastandsignaal
AND_OP           ; Kahe signaali vaheline loogiline 'ja'
BEZ l_else       ; Kui 'and' ei kehti, siis hüppa märgise l_else juurde
```

```
PUSHL 3          ; Pesa C
PUSHL 1          ; Signaal 1
DWR              ; Väljasta pesast C signaal 1
BRA l_end        ; Hüppa lõpu märgise juurde
LABEL l_else     ; Märgi else algus
PUSHL 3          ; Pesa C
PUSHL 0          ; Signaal 0
DWR              ; Väljasta pesast C signaal 1
LABEL l_end      ; Märgi kogu if-else lause lõpp
STOP
```

Joonis 33: **if-else** konstruktsioon Crumble'i masinkoodina

Crumblepy's kasutab **if**-lause märgendeid, et vastavalt tingimuse kehtimisele hüpata kas **if**-haru või **else**-haru sisse. Tingimuseks võivad olla kõigi funktsioonide tagastusväärtused. Tõeseks väärtuseks loetakse kõike, mis ei ole 0. Vääraks väärtuseks loetakse 0. Samuti on võimalik kasutada loogilisi konstante `True` ja `False`, mis aga Crumble masinkoodis loogilise tüübi puudumise tõttu kompileeritakse vastavalt arvudeks 1 ja 0.

2.3.3 Tsüklid

Crumblepys on võimalik kasutada vaid **while**-tsüklit. Toetatud on ka **continue** ja **break** laused vastavalt tsükli järgmisele iteratsioonile hüppamiseks ning tsükli lõpetamiseks. Pythoni **for**-tsüklit ei ole võimalik kasutada Crumble masinkoodis `list` tüübi puudumise tõttu.

2.3.4 Muutujad

Muutujate väärtustamine ja kasutamine toimub Crumblepys identselt Pythonile. Joonisel 34 on näidatud Crumblepys muutujale väärtuse andmine.

```
a_and_b = get_digital(A) and get_digital(B)
```

Joonis 34: Muutuja väärtustamine Crumblepys

Erinevus Pythonist seisneb vaid võimalikes muutujatüüpides. Võimalik on kasutada vaid täisarve ja tõeväärtusi (`int` ja `bool`). Kusjuures tõeväärtused kompileeritakse täisarvudeks 1 ja 0 (vastavalt `True` ja `False`). See tähendab, et näiteks `True + 3` Crumblepy koodis on võrdne arvuga 4.
Samuti on reserveeritud neli muutujat `A`, `B`, `C` ja `D` vastavalt täisarvuliste väärtustega 0, 1, 3 ja 2, et oleks võimalik lihtsamalt viidata pesadele funktsioonides, mis vajavad pesa identifikaatoreid. Neid muutujaid ei ole lubatud üle kirjutada. See

toob kaasa kompileerimisel veateate. Pesade identifikaatoritena on võimalik kasutada ka nende muutujate endi täisarvulisi väärtusi. Järgmised kaks koodijuppi on seega võrdväärsed: `get_digital(A)` ja `get_digital(0)`.

2.4 Kompilaatori kasutamine

Crumblepy kompilaator on loodud olema käsurealt kasutatav. Käsurea parsimiseks kasutatakse `argparse` moodulit. Parsimine toimub failis „crumblepy/__init__.py" Kompilaatori käsureaargumendid on nähtavad, kirjutades käsureal `python crumblepy -h`. Selle käsu tulemus on näidatud joonisel 35.

```
> python crumblepy -h

usage: crumblepy [-h] [-o OUTPUT] [-f {bytecode,assembly,usb}]
                 file

Compile a .crpy source file

positional arguments:
  file                  The .crpy file to compile

optional arguments:
  -h, --help            show this help message and exit
  --output OUTPUT, -o OUTPUT
                        File to save output to. Use --format to
                        specify format. If unspecified, output
                        is instead printed to console
  --format {bytecode,assembly,usb}, -f {bytecode,assembly,usb}
                        Output format. Defaults to usb.
```

Joonis 35: Crumblepy kompilaatori käsureaargumendid

Käsul on üks kohustuslik argument `file`, mis määrab Crumblepy lähtekoodi faili, mida kompilaator peab kompileerima. Võimalik on määrata argumendiga `--format` väljundi tüüp. Väljundi tüüp `bytecode` väljastab koodi baitide nimekirjana, `assembly` väljastab koodi assemblerkoodina, `usb` saadab kompileeritud masinkoodi USB ühenduse kaudu Crumble kontrollerile. Kasutades argumenti `--output`, on võimalik määrata, et konsooli trükkimise asemel salvestataks väljund faili. Kui on määratud `--format usb` (mis on ka vaikeväärtuseks), siis `--output` ei tee midagi.

2.5 Testimine

Crumblepy kompilaatori testimiseks on loodud programmid, mille eesmärgiks on testida kõiki kompilaatori funktsionaalsusi. Need programmid asuvad kaustas „test/crpy_sources". Iga programmi päises on kommentaar testitava funktsionaalsuse kirjelduse ning oodatud tulemusega. Testi jooksutamiseks on tarvis Crumble ning sobivad Crumb'id ja Sparkle'id vastavalt testi kirjeldusele ja oodatud tulemusele omavahel ühendada ning Crumble kontroller USB kaudu arvutiga ühendada. Peale seda tuleks kompileerida testskript ning see saata USB kaudu Crumble kontrollerile.

```
python crumblepy test/crpy_sources/d_enable.crpy
```

Joonis 36: Testskripti kompileerimine ja Crumble kontrollerile saatmine

Joonisel 36 on näidisena toodud testskripti „test/crpy_sources/d_enable.crpy" kompileerimine ning kontrollerile saatmine Crumblepy kompilaatori käsu abil. Kui Crumble käitub vastavalt testi päises toodud oodatud tulemusele, siis on test edukalt läbitud.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli luua Crumble kontrollerile programmeerimiskeel, mis põhineb keelel Python. Töös anti ülevaade Crumble kontrollerist, sellega ühilduvatest komponentidest ning Crumble'i arenduskeskkonnast ja -protsessist. Loodi Pythoni baasil programmeerimiskeel Crumblepy, millega on võimalik Crumble kontrollerit juhtida, ning anti ülevaade selle loomisprotsessist. Samuti kirjeldati Crumblepy'ga töötamist: keele funktsionaalsusi ning selle kompileerimist.
Valminud programmeerimiskeel ja selle kompilaator võimaldavad Pythoni-sarnase keelega Crumble kontrollerit juhtida. Sarnasus Pythoniga toetab algaja Pythonis programmeerija edasisi õpinguid paremini kui Crumble'i enda arenduskeskkond. Käesolevat tööd oleks võimalik edasi arendada, luues valiku õppematerjale või projekte, mida Crumblepy'd kasutades oleks võimalik koolitundides kasutada.

Viidatud kirjandus

[1] *The Crumble Controller | Redfern Electronics*. http://redfernelectronics.co.uk/crumble/ (04. 03. 2017).

[2] *4tronix*. http://www.4tronix.co.uk/store/ (04. 03. 2017).

[3] *Basic Connections*. http://redfernelectronics.co.uk/wp-content/uploads/2014/05/Basic-Connections-595x275.png (04. 03. 2017).

[4] *Scratch - Imagine, Program, Share*. https://scratch.mit.edu/ (04. 03. 2017).

[5] *Crumble Friendly Components*. http://redfernelectronics.co.uk/wp-content/uploads/2017/01/IMG_0115-e1485789913691.jpg (04. 03. 2017).

[6] *Crumble-Friendly Components (Pack) | Redfern Electronics*. http://redfernelectronics.co.uk/product/crumble-friendly-components-pack/ (04. 03. 2017).

[7] *Snap Action Switches (or Microswitch) Are Rapid Switch Devices for an Electrical Circuit - Future Electronics*. http://www.futureelectronics.com/en/switches/snap-acting-switches.aspx (04. 03. 2017).

[8] Chris Woodford. *How Reed Switches Work (Magnetically Operated Switches)*. 14. jaanuar 2017. http://www.explainthatstuff.com/howreedswitcheswork.html (04. 03. 2017).

[9] *HC-SR04 Back with Crumbliser*. http://redfernelectronics.co.uk/wp-content/uploads/2016/04/HC-SR04-back-with-Crumbliser.jpg (04. 03. 2017).

[10] *Ultrasonic Distance Measuring Sensor | Redfern Electronics*. http://redfernelectronics.co.uk/product/ultrasonic-distance-measuring-sensor/ (04. 03. 2017).

[11] *HC-SR04 User's_Manual*. https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit?usp=embed_facebook (04. 03. 2017).

[12] *Line Follower Board*. http://redfernelectronics.co.uk/wp-content/uploads/2015/09/Line-follower-board.jpg (04. 03. 2017).

[13] *Two Sparkles to Crumble*. http://redfernelectronics.co.uk/wp-content/uploads/2014/05/Two-Sparkles-to-Crumble.png (04. 03. 2017).

[14] *Sparkles (Panel of 25) | Redfern Electronics*. http://redfernelectronics.co.uk/product/sparkles-panel-of-25/ (04. 03. 2017).

[15] Worldsemi. *WS2812B*. https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf (04. 03. 2017).

[16] *Crumblizer with WS2812B Strip*. http://redfernelectronics.co.uk/wp-content/uploads/2015/10/Crumblizer-with-WS2812B-Strip.jpg (04. 03. 2017).

[17] *Crumbliser (Pack of 5) | Redfern Electronics*. http://redfernelectronics.co.uk/product/crumbliser-pack-of-5/ (04. 03. 2017).

[18] *Pair-of-Geared-Motors*. http://redfernelectronics.co.uk/wp-content/uploads/2015/10/Pair-of-geared-motors-300x300@2x.jpg (04. 03. 2017).

[19] *Pair of Geared Motors | Redfern Electronics*. http://redfernelectronics.co.uk/product/pair-of-geared-motors/ (04. 03. 2017).

[20] *Crumble*. http://4tronix.co.uk/store/index.php?rt=product/category&path=82 (04. 03. 2017).

[21] *4tronix Toodetavad Crumb'id*. http://4tronix.co.uk/store/resources/image/18/b7/8.jpg (04. 03. 2017).

[22] *Kiirendusandur [Robotic & Microcontroller Educational Knowledgepage - Network of Excellence]*. http://home.roboticlab.eu/et/examples/sensor/accelerometer (11. 03. 2017).

[23] *Motion Sensor Crumb Accelerometer for Crumble Controller*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=480 (11. 03. 2017).

[24] *Motion Sensor Crumb Accelerometer for Crumble Controller*. http://4tronix.co.uk/store/resources/image/18/b9/4.jpg (11. 03. 2017).

[25] *Proximity Sensor Crumb Digital Input for Crumble Controller*. http://4tronix.co.uk/store/resources/image/18/bd/8.jpg (11. 03. 2017).

[26] *Proximity Sensor Crumb Digital Input for Crumble Controller*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=494 (11. 03. 2017).

[27] *Dual Touch Sensor Crumb Digital Input for Crumble Controller*. http://4tronix.co.uk/store/resources/image/18/bc/b.jpg (11. 03. 2017).

[28] *Dual Touch Sensor Connected to Fruit*. http://4tronix.co.uk/store/resources/image/18/bc/d.jpg (11. 03. 2017).

[29] *Number Display Crumb for Crumble Controller*. http://4tronix.co.uk/store/resources/image/18/bc/8.jpg (11. 03. 2017).

[30] *Number Display Crumb for Crumble Controller*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=479 (11. 03. 2017).

[31] *Passiivne Infrapunasensor*. http://4tronix.co.uk/store/resources/image/18/be/0.jpg (11. 03. 2017).

[32] *PIR Sensor Crumb Digital Input for Crumble Controller*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=495 (11. 03. 2017).

[33] *Servo Crumb for Crumble Controller*. http://4tronix.co.uk/store/resources/image/18/ca/8.jpg (11. 03. 2017).

[34] *Servomootor [Robotic & Microcontroller Educational Knowledgepage - Network of Excellence]*. http://home.roboticlab.eu/et/examples/motor/servo (11. 03. 2017).

[35] *Servo Crumb for Crumble Controller*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=503 (11. 03. 2017).

[36] *Playground for Crumble & BBC Micro:Bit*. http://4tronix.co.uk/playground/ (11. 03. 2017).

[37] *PlayGround, Gizmos and Wires*. http://4tronix.co.uk/blog/wp-content/uploads/2016/09/PlayCrum_03.jpg (11. 03. 2017).

[38] *Prototyping Gizmo for Playground - Make Your Own Gizmo*. http://4tronix.co.uk/store/index.php?rt=product/product&path=82&product_id=576 (11. 03. 2017).

[39] *PIC16F1455 - Microcontrollers and Processors*. http://www.microchip.com/wwwproducts/en/PIC16F1455 (04. 03. 2017).

[40] *Python2orPython3 - Python Wiki*. https://wiki.python.org/moin/Python2orPython3 (26. 03. 2017).

[41] Sebastian Raschka. *The Key Differences between Python 2.7.x and Python 3.x with Examples*. 1.–juuni 2014. http://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html (26. 03. 2017).

[42] *32.2. Ast — Abstract Syntax Trees — Python 3.6.1 Documentation*. https://docs.python.org/3/library/ast.html (26. 03. 2017).

Lisad

1 Crumble kontrolleri masinkoodi käsud

Käsk	Kood	Pinu argumendid	Kirjeldus
`PUSH` a	`00 00`	-	Paneb pinusse muutuja nr a väärtuse
`PUSHL` a	`01 00`	-	Paneb pinusse väärtuse a
`POP` a	`03 00`	p	Seab muutuja number a väärtuseks p
`DUP`	`06 00`	p	Paneb pinusse $p\ p$ (kloonib pinu pealmise väärtuse)
`ADD`	`08 00`	$p_1\ p_2$	Paneb pinusse $p_1 + p_2$ tulemuse
`SUB`	`09 00`	$p_1\ p_2$	Paneb pinusse $p_1 - p_2$ tulemuse
`MUL`	`0A 00`	$p_1\ p_2$	Paneb pinusse $p_1 * p_2$ tulemuse
`DIV`	`0B 00`	$p_1\ p_2$	Paneb pinusse p_1/p_2 tulemuse
`GTE`	`10 00`	$p_1\ p_2$	Paneb pinusse $p_1 \geq p_2$ tulemuse
`GT`	`11 00`	$p_1\ p_2$	Paneb pinusse $p_1 > p_2$ tulemuse
`LTE`	`12 00`	$p_1\ p_2$	Paneb pinusse $p_1 \leq p_2$ tulemuse
`LT`	`13 00`	$p_1\ p_2$	Paneb pinusse $p_1 < p_2$ tulemuse
`EQ`	`14 00`	$p_1\ p_2$	Paneb pinusse $p_1 == p_2$ tulemuse
`RND`	`0C 00`	$p_1\ p_2$	Paneb pinusse suvalise täisarvu p_1 ja p_2 vahel
`BRA` a	`18 00`	-	Hüppab masinkoodi reale nr a
`BEZ` a	`19 00`	p	Hüppab masinkoodi reale nr a, kui $p \neq 1$
`AND`	`20 00`	$p_1\ p_2$	Paneb pinusse $p_1 \wedge p_2$ tulemuse
`NOT`	`22 00`	p	Paneb pinusse $\neg p_1$ tulemuse
`OR`	`23 00`	$p_1\ p_2$	Paneb pinusse $p_1 \vee p_2$ tulemuse
`DRD`	`28 00`	p	Paneb pinusse pesast p loetud digitaalse väärtuse (1 või 0)
`DWR`	`29 00`	$p_1\ p_2$	Väljastab pesast p_1 digitaalse signaali p_2
`ARD`	`2A 00`	p	Paneb pinusse pesast p loetud analoog-väärtuse
`MOT1`	`30 00`	p	Paneb mootori 1 liikuma kiirusega p
`MOT2`	`31 00`	p	Paneb mootori 2 liikuma kiirusega p
`SSPRK`	`32 00`	$p_1\ p_2\ p_3\ p_4$	Seab Sparkle'i number p_1 värviks RGB koodiga värvi p_2, p_3, p_4
`WAIT`	`34 00`	p	Peatab programmi üheks millisekundiks
`SRV`	`26 00`	$p_1\ p_2$	Seab pesaga p_1 ühendatud servomootori nurgaks p_2 kraadi
`STOP`	`3F FF`	-	Peatab programmi töö

2 Litsents

Mina, **Rasmus Saks**

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „**Crumble kontrollerile Pythoni-laadse programmeerimiskeele loomine**", mille juhendaja on **Aivar Annamaa**

    1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni

    1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace'i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **10. mai 2017. a.**